11

Controller de autenticación

Transcripción

[00:00] Ya vimos cómo configurar Spring para que utilice el tipo de autenticación stateless, ahora es lo que nos está faltando aquí es ver nuestro método de login. Para esto voy a copiar esta request, lo voy a duplicar mejor dicho, pero aquí le voy a decir login usuario.

[00:25] Entonces head aquí, le aquí vamos a crear y en este login vamos a decirle que va a ser en el endpoint login y va a ser un método del tipo post. Perfecto. En la parte del body yo le voy a agregar un JSON con lo que yo quiero que tenga mi método de login, por ejemplo, mi login va a tener exactamente igual que lo definí en mi interfaz, va a tener "login", va a ser mi usuario, vamos a hacer de cuenta que es el mío.

[00:54] "diego.rojas", y mi clave que va a ser digamos del 1 al 6, mi clave del 1 al 6. Y listo, esto sería todo por ahora. Si yo disparo este método lo más seguro es que me retornen un 404, porque no existe aún, vamos a crearlo. Para esto vamos a cerrar un poco esto, venimos a nuestro controller y vamos a crear aquí nuevo AutenticaciónController.

[01:36] Vamos a recordar un poco de cómo configurar un controller. Lo primero que tengo que hacer es decirle que es un @RestController, tengo que hacer @RequestMapping, ¿con qué va a ser el @RequestMapping? Con "/login" porque estamos apuntando aquí a este path, a este recurso y ahora vamos a tener nuestro método de login, por ejemplo, ya vimos que en lugar de retornar void, vamos a retornar ResponseEntity.

[02:01] Entonces vamos a decirle public ResponseEntity login autenticación usuario mejor, autenticarUsuario y listo, eso debía retornar algo. Perfecto. Este va hacer un @PostMapping y esto es solo un repaso de lo que hemos visto en el curso anterior y como parámetro yo voy a recibir un body, como estoy diciendo aquí, un body con un login y una clave.

[02:34] Si yo voy a recibir parámetros de mi cliente ¿qué necesito? Necesito un DTO, perfecto. Entonces le voy a decir aquí datos autenticación usuario y obviamente me va a dar un error de compilación porque no existe, lo que voy a hacer es crear mi récord como ya lo sabemos y lo voy a crear en mi paquete usuario.

[03:00] Vamos a decirle que pase a nivel de usuario. Déjame buscar aquí, me voy a crear aquí, no se ocupen y después le voy a cambiar aquí, va a ser med.voll.api.domain.usuarios. Va a ir aquí. Y ahora lo que voy a hacer aquí es moverlo al paquete usuarios y listo, ya está.

[03:28] Entonces si ven aquí mi dominio usuarios ya está mi nuevo DTO. En este DTO yo le voy a declarar los parámetros string login, string clave. Y eso sería todo con respecto a mi usuario. Aquí ya tengo mi DTO, datosAutenticacionUsuario, ya está listo. ¿Qué más sigue ahora? Para digamos disparar el proceso de autenticación en Spring existe una clase en específico que inicia todo este proceso. Esta es el autenticationManager.

[04:05] ¿Cómo funciona? Declaramos aquí como para un servicio private authenticationManager de Spring, recuerden con @Autowired vamos a inyectarlo, Spring la debe tener en su contexto en alguna parte y como está inyectando su authenticationManager, y aquí lo que yo le voy a decir es autenthicationManager.authenticate.

[04:34] Y aquí él está esperando algo. Está esperando un objeto del tipo authentication. ¿Y este objeto qué va a ser? Va a ser nada más ni nada menos que mi token que yo tengo aquí en mi método de login.

[04:54] Entonces lo que yo voy a crear aquí va a ser una variable, digamos, token, que no existe obviamente, entonces voy a crear esta variable local aquí y va a ser el token. Y perfecto, con esto ya debería ser suficiente. Como toquen es una interfaz, lo que yo tengo que hacer aquí es usar una implementación de esta interfaz, la cual es la primera que nos sale aquí, UsernamePasswordAuthenticationToken.

[05:25] Y para esto él me va a pedir dos cosas aquí. Me va a pedir mis datosUsuario.login y mis datosAutenticacionUsuario.clave. Nada más, es lo único que necesito, UsernamePasswordAuthenticationToken con mis datos de autenticación de usuario: en este caso mi login y mi clave, nada más. ¿Ahora qué tengo que hacer?

[05:52] Solamente para fines didácticos lo que voy a hacer aquí es retornar un ResponseEntity.ok() Listo y con esto sale un error aquí, porque no está como ResponseEntity, a ver, este ResponseEntity salió de otra parte pareces. Vamos a ver de dónde salió ResponseEntity, y el otro. Faltaba el build. Listo, perfecto.

[06:20] Y listo, eso sería todo lo que yo necesito por ahora para hacer ese trigger digamos de alguna forma de mi método de login generando con mis datosAutenticacionUsuario, un token que es este de aquí que va a ser usado por AuthenticationManager para autenticar. Entonces voy a mi servidor.

[06:54] Vemos que aquí falló porque aquí no estaba compilando por un tema que no está consiguiendo hacer una inyección. Y lo que me dice aquí es el campo authenticationManager en la clase authenticationController requiere un objeto del tipo AuthenticationManager que no puede ser encontrado. ¿Qué significa esto?

[07:13] Que si yo quiero inyectar un objeto del tipo AuthenticationManager que Spring no lo tiene su contexto entonces mi código no va a compilar. El IDE ya me estaba avisando sobre este error, yo lo ignoré intencionalmente para poder

mostrarles este tipo de error que sale cuando tienes un bean declarado en tu contexto.

[07:33] ¿Qué tenemos que hacer entonces? Vamos a nuestro paquete de infra, a nuestro paquete security, y en nuestra configuración si nos está pidiendo un objeto del tipo AuthenticatinManager tenemos que dárselo a Spring. Entonces lo que voy a hacer es venir aquí a mis configuraciones y de la misma forma que declaro mi SecurityFilterChain, yo voy a dar aquí un public AuthenticationManager de Sprint, authenticationManager, y esto me va a retornar, vamos a ver.

[08:09] Estoy recibe como parámetro un AuthenticationConfiguration. Vamos a dejarlo como AuthenticationConfiguration y lo que le vamos a decir es que retorne AuthenticationConfiguration.getAuthenticationManager. Acá nos va a pedir que lancemos una excepción, la dejamos aquí en la firma el método, le damos un enter para que sea más legible y lo único que falta para que Spring la tenga en su contexto es anotarlo con @Bean.

[08:45] Listo. Ya tenemos entonces el AuthenticationManager, nuestro SecurityFilterChain, entonces borramos esto y reiniciamos nuestro servidor. Vamos a ver qué sale ahora. Digo que la aplicación inicializó, vamos a Insomnia, como este endpoint ya existe, deberíamos hacer algo, vamos a enviar y me sale 403 forbidden. ¿Qué es lo que sucedió aquí?

[09:17] Y en efecto miren aquí, vamos a agrandar esto un poco más. Vimos que ya se hizo un query en la base de datos. ¿Esto qué quiere decir? Que satisfactoriamente todo mi proceso que he hecho, todos mis beans, mis declaraciones que he hecho han ido por mi controller, déjenme llegar a controller. Han venido por el controller, por aquí, en efecto entró aquí en UsernamePasswordAuthenticationToken, intentó encontrar mi usuario y mi clave en mi base de datos.

[09:50] O sea, llegó desde el controller, hasta mi repository, intentó encontrarlo por login, ¿pero qué pasó? No encontró nada. ¿Por qué? Porque ese usuario y esa clave no existen aún en mi base de datos. Entonces yo estoy preguntando por un usuario que es inexistente. En el siguiente video ya vamos a aprender cómo guardar este usuario en la base de datos y generar ya nuestros tokens JWT. Nos vemos.